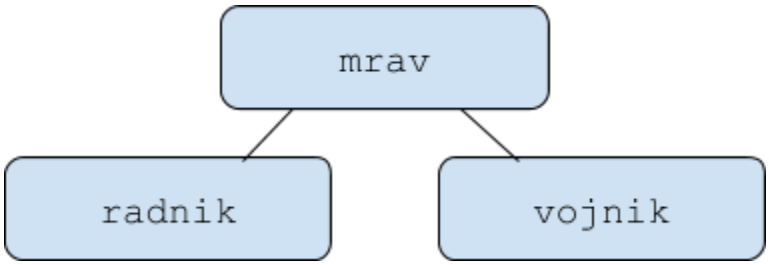
Zadatak

Mravinjak se nalazi u ishodištu cjelobrojnog koordinatnog sustava. U mravinjaku nastaju mravi koje mogu dosegnuti sve točke unutar kvadrata kojemu je donji lijevi kut (-50, -50) a gornji desni (50,50).
Radnik sakuplja hranu i ima snagu=1, vojnik čuva od opasnosti i ima snagu=2. Svi mravi se kreću samo koracima gore, dolje, lijevo, desno, i svi imaju brzinu 1 korak/jedinici vremena.
Za udaljenost neke dvije točke uzimamo najmanji broj koraka potreban da se iz jedne dospije u drugu. (na primjer, udaljenost((1,2),(4,-7))=12)

Napravite sljedeću hijerarhiju klasa koja opisuje njihovo ponašanje.

Sučelje spremite u datoteku **mrav.h**, a implementaciju spremite u datoteku **mrav.cpp.** Sve funkcije i strukture smiju biti javne.



Mrav neka je apstraktna klasa. Implementirajte sljedeće funkcije članice:

- svi mravi dijele statičke varijable `int Opasnost` (koja je defaultno nula, opisuje opasnost na koordinatama (0,0)) i `int zaliha` (koja je defaultno =20).
- konstruktori – stvaraju novog mrava/vojnika/radnika **u** mravinjaku, umanjuje zalihu za 1. U test primjeru neće biti pozivanja konstruktora sa zalihom 0.
- `int gdjeX(), int gdjeY(), presao()` - vraćaju trenutne koordinate mrava, te koliko koraka je presao od nastanka.
- `string oznaka()` - vraća "`vojnik`" ili "`radnik`"
- `mrav& najblizi()` - vraća referencu na najbližeg mrava, (obratite pozornost kako je definirana udaljenost). Ako ima više kandidata, onoga koji je ranije kreiran. Neće biti pozivana u situaciji gdje postoji samo jedan mrav.
- `pomak`**`Za`**`(int dx, int dy)` – funkcija članica u klasi `mrav`. Instruira ga da se krene pomjerati **za** (dx,dy), i to ovisno o vrsti mrava: `radnika` da se kreće za dx po x osi pa za dy po y osi. `vojniku` pak koordinate mijenja obratnim redom. Ako dođe do ruba, prestaje kretanje po toj koordinati, prelazi na eventualnu iduću koordinatu. Dakle za **`radnika`** A koji je u ishodištu, poziv `A.pomak(51,1)` ga usmjeri da za 50+1 vremensku jedinicu preseli A na koordinate (50,1). Neka je A dospio do (50,1). Novi poziv A.pomakZa(-1,**-53**) usmjerava A na polje (49,**-50**) (prvo 1 korak lijevo pa **51** dolje, jer bi sa 53 dolje ispao sa "ploče").
- `void sakupljaj(int x, int y)` - funkcija članica za `radnika`. Ako je već nosio hranu, od nekog ranijeg poziva funkcije sakupljaj, zaputi se u mravinjak da je odloži pa ide prema koordinatama (x,y), uzme jednu jedinicu hrane, i otamo nosi hranu prema mravinjaku. Ako nije ništa nosio, zaputi se odmah na (x,y), pa od tamo nosi hranu u mravinjak. **Radnik se kreće po istim pravilima kao u pomakZa (prvo se mijenja x koordinata pa y koordinata).**

Kad radnik donese hranu u mravinjak, zaliha mravinjaka se uveća za 1. Dok ne dobije drugu instrukciju, mrav nastavlja sakupljati hranu sa polja (x,y).

- `void patroliraj(int x, int y)` – funkcija `vojnika`. Instruira ga da patrolira po rubu pravokutnika (0,0)->(0,y)->(x,y)->(x,0)->(0,0), dok ne dobije drugačije instrukcije. **Vojnik se kreće po istim pravilima kao u pomakZa (prvo se mijenja y koordinata pa x koordinata).** Ako u trenu poziva funkcije vojnik nije u ishodištu, plan puta mu je prvo točka (x,y), pa onda obilazak (x,y)->(x,0)->(0,0)->(0,y)->.
- `opasnost(int k=1) - static` funkcija koja poveća `Opasnost` za k.
- `Svi() - static` funkcija koja vraća listu pointera na sve postojeće mrave, po redu nastanka.
- `vrijeme(int k=1)` – `static` funkcija, simulira protok k vremenskih jedinica. U jednoj vremenskoj jednici se desi slijedeće:
  Neka je s=snaga svih mrava trenutno u mravinjaku.(1) Ako je s>=`Opasnost`, `Opasnost` postane 0, i **svi** mravi nastavljaju sa svojim prethodnim zadacima.
  (2) Ako `Opasnost>s`, mravi koji su trenutno u mravinjaku ostanu tu vremensku jedinicu u mravinjaku i smanje opasnost za s. Ostali mravi nastavljaju sa svojim zadacima.

Možete staviti `public` pristup u svim klasama za sve varijable i funkcije. Više mrava se može nalaziti na istoj točki. Ako za funkciju nije napomenuto da je specifična za izvedenu klasu, znači da se odnosi na baznu klasu. Gore navedeni su **nužni** elementi strukture, smijete imati i druge dodatne. Mrav pamti samo zadnji nalog koji mu je zadan (sakupljaj, patroliraj ili pomak).

Opće napomene

- Struktura, funkcije i datoteke moraju se zvati točno onako kako je zadano u zadatku. Pazite na mala i velika slova.
- Trebate poslati samo sučelje i implementaciju. U datoteci koju šaljete ne smije se nalaziti funkcija main()!
- Nijedna funkcija ne smije ništa učitavati s tipkovnice ili neke datoteke, niti išta ispisivati na ekran ili neku datoteku.
- Svaki od mainova kojim testiramo vaše programe neće pozivati sve navedene funkcije. Stoga, ako neku funkciju ne znate napisati, ipak možete skupiti koji bod na funkcije koje znate implementirati (u tom slučaju funkciju koju ne znate nemojte navesti u .h datoteci ili napravite neku trivijalnu implementaciju)

Ispravnost implementacija koje napišete bit će provjerena tako da ćemo mi napisati razne klijentske programe koji će deklarirati nekoliko varijabli zadane strukture, i na njima pozivati funkcije koje ste trebali napisati. Ako se poslani programi ne budu uspješno povezivali (linkali) s našim klijentskim programima, smatrat će se neispravnima.

Neki klijentski programi provjeravat će samo neke jednostavnije funkcije, dok će neki provjeravati sve funkcije koje trebate napisati. Provjera je potpuno automatska, tako da je od presudne važnosti da se pridržavate specifikacije. Nepridržavanje lako može uzrokovati osvojenih 0 bodova iz zadaće!

Upiti u vezi zadaće na mail *marijan.polic@math.hr*

Primjer maina:

```
#include<iostream>
#include<list>
#include”mrav.h”

using namespace std;
int main() {
	radnik r1, r2, r3;
	vojnik v1, v2, v3;
	list<mrav*>  S;
	list<mrav*> ::iterator i;

	v1.patroliraj(34,45); v2.patroliraj(-7,-7);
	r1.pomakZa(39,2); r2.sakupljaj(-7,10); r3.sakupljaj(1,1);
	S=mrav::Svi();

	mrav::vrijeme(5);
	for(i=S.begin(); i!=S.end(); ++i) cout<<(*i)->oznaka()<<" na
"<<(*i)->gdjeX()<<","<<(*i)->gdjeY()<<endl;/*
radnik na 5,0
radnik na -5,0
radnik na 1,0
vojnik na 0,5
vojnik na 0,-5
vojnik na 0,0 */

	cout<<endl;
	mrav::opasnost(10);
	mrav::vrijeme(3);
	for(i=S.begin(); i!=S.end(); ++i) cout<<(*i)->oznaka()<<" na
"<<(*i)->gdjeX()<<","<<(*i)->gdjeY()<<endl;
/*radnik na 8,0
radnik na -7,1
radnik na 0,0         //ovaj radnik ce zastati u 4. Vrem. jedinici pomoci v3 sa Opasnosti
vojnik na 0,8
vojnik na -1,-7
vojnik na 0,0
*/

	v1.vrijeme(70);cout<<endl;
	for(i=S.begin(); i!=S.end(); ++i) cout<<(*i)->oznaka()<<" na
"<<(*i)->gdjeX()<<","<<(*i)->gdjeY()<<endl;/*
radnik na 39,2
radnik na -7,3
radnik na 1,0
vojnik na 33,45
vojnik na -6,0
vojnik na 0,0
*/

	mrav& pom=v1.najblizi();cout<<endl;
	cout<<pom.oznaka()<<" na "<<pom.gdjeX()<<","<<pom.gdjeY()<<endl;  //radnik na 39,2

}
```